JOURNAL OF COMPUTER AND SYSTEM SCIENCES **54**, 177–189 (1997)
ARTICLE NO. SS971478

A Weak Version of the Blum, Shub, and Smale Model

Pascal Koiran*

Laboratoire de l'Informatique du Parallélisme, École Normale Supérieure de Lyon—CNRS, France

Received February 12, 1995; revised February 7, 1996

We propose a weak version of the Blum–Shub–Smale model of computation over the real numbers. In this weak model only a "moderate" usage of multiplications and divisions is allowed. The class of boolean languages recognizable in polynomial time is shown to be the complexity class P/poly. The main tool is a result on the existence of small rational points in semi-algebraic sets which is of independent interest. As an application, we generalize recent results of Siegelmann and Sontag on recurrent neural networks, and of Maass on feedforward nets. A preliminary version of this paper was presented at the *1993 IEEE Symposium on Foundations of Computer Science*. Additional results include: an efficient simulation of order-free real Turing machines by probabilistic Turing machines in the full Blum–Shub–Smale model; the strict inclusion of the real polynomial hierarchy in weak exponential time.

1. INTRODUCTION

The real number model of computation is a quite natural and convenient framework for studying numerical computation. It is nonetheless often criticized for being overly unrealistic. The assumption that arbitrary real numbers can be compared, added, or multiplied in constant time and infinite precision is the usual target of these criticisms (see [23] for a discussion of this issue). This paper proposes a model of computation (the "weak BSS model") which can be seen as a middle ground between the usual Turing machine model and the real Turing machine of Blum, Shub, and Smale [4].

The weak BSS model is obtained from the standard one by dropping the constant cost hypothesis for arithmetic operations. This yields a notion of feasible analog computation which is perhaps more realistic, since the standard notion of discrete polynomial-time computability is recovered when the real parameters defining a real Turing machine happen to be rational. In other words, an algorithm that runs in polynomial time in the weak model is "ideal" in the sense of Renegar, since "it seems that the ideal algorithm is one that provides a 'nice' bound in the real number model and which when restricted to sentences with polynomials whose coefficients are rational number also provides a 'nice' bound in the bit model" [18].

One important theme of this paper is that when both input and outputs of real machines are discrete, complexity classes in the real number model can be identified to discrete complexity classes. For instance, our main theorem states that the class of boolean languages recognizable in polynomial time in the weak BSS model is P/poly (P/poly is the class of sets recognized in polynomial time by Turing machines using advice functions of polynomial length or, equivalently, by nonuniform boolean circuits of polynomial size [1]). The main tool for this proof is a new result on semialgebraic sets of independent interest, stating that a semialgebraic set of nonempty interior defined by "small" equations must contain a "small" rational point. It follows from this characterization that $P \neq NP$ in the weak model under a standard complexity-theoretic assumption, even if boolean (0/1) guesses only are allowed. After a first version of this paper was written [13], it was shown in [10] by quite different methods that for full (real) nondeterminism, $P \neq NP$ holds in the weak model without any assumption.

The rest of this paper is organized as follows. After recalling some standard background on semialgebraic sets in Section 2, we present the full and weak BSS models in Section 3. The main theorem is stated here, along with simpler results of same type. The existence of small points in semialgebraic sets is established in Section 4; as explained in Section 5, the main theorem follows from this result. Models of computation without order are studied in Section 6; it is shown that real Turing machines can be efficiently simulated by ordinary Turing machines in the weak model and by probabilistic Turing machines in the full BSS model. In Section 7, we propose a real version of the Meyer–Stockmeyer polynomial hierarchy and show that it is strictly included in weak exponential time. In particular, the inclusion of nondeterministic polynomial time in weak exponential time is strict. This was left as an open problem in [10]. We give applications of our results to recurrent and

* A part of this work was done while the author was visiting DIMACS at Rutgers University. Support by an INRIA fellowship and Esprit Project 8556 (NeuroCOLT) is also acknowledged.

E-mail: koiran@lip.ens-lyon.fr

0022-0000/97 $25.00

feedforward neural networks in Section 8. It is shown that that the class of functions computed by recurrent nets using (possibly discontinuous) piecewise-linear output functions is exactly P/poly. This was previously known only for continuous output functions [22]. We also show that a family of feedforward nets of logarithmic (and sometimes unbounded) depth can be simulated efficiently by a family of threshold circuits if it uses only a constant number of real parameters. This complements a result of Maass [16] who showed a similar result for circuits of constant depth using an unbounded number of parameters.

2. PRELIMINARIES

In this section, we list the notations used throughout the paper and recall some standard background on semialgebraic sets (a comprehensive treatment of this topic can be found in [3, 5]).

2.1. *Miscellaneous Notations*

The set $\mathbb{N}$ of nonnegative integers is identified with $\{0, 1\}^+$; the set of nonempty words on the alphabet $\{0, 1\}$.

The size $s(x)$ of an integer $x \in \mathbb{Z}$ is the number of digits of its radix-2 representation. The size of a rational number $x \in \mathbb{Q}$ is $s(x) = \min\{s(p) + s(q), x = p/q, p, q \in \mathbb{Z}\}$. The size of a rational point $x = (x_1, ..., x_p) \in \mathbb{Q}^p$ is $s(x) = \sum_{i=1}^{p} s(x_i)$.

Given a ring A, $A[X_1, ..., X_p]$ is the ring of polynomials in p variables with coefficients in A. When there is no risk of confusion, we write also $A[X]$, instead of $A[X_1, ..., X_p]$. The degree of $P \in A[X_1, ..., X_p]$ is $d(P)$, and the degree of a rational fraction $R = P/Q$ is $d(R) = d(P) + d(Q)$.

2.2. *Semialgebraic Sets*

DEFINITION 1. A basic semialgebraic set $\mathscr{S} \subseteq \mathbb{R}^p$ is defined by a conjunction of polynomials equalities and inequalities:

$$\mathscr{S} = \{x \in \mathbb{R}^p;\ Q_1 > 0 \wedge \cdots \wedge Q_m > 0 \wedge R_1 = 0 \wedge \cdots \wedge R_n = 0\},$$

where $Q_i, R_i \in \mathbb{Z}[X_1, ..., X_p]$ are nonconstant polynomials (m and/or n may be equal to zero). A semialgebraic set is a finite union of basic semialgebraic set.

Usually all sign conditions ($P = 0$, $P \leqslant 0$, $P < 0$, $P \geqslant 0$, and $P > 0$) are allowed in the definition of a semialgebraic set, but these two definitions are clearly equivalent.

It is possible to take the P_i's and R_i's in $\mathbb{R}[X]$ instead of $\mathbb{Z}[X]$. The resulting sets will also be called semialgebraic. It should be clear from the context whether the polynomials have integer or real coefficients.

THEOREM 1 (The Tarski–Seidenberg principle). *Let $S' = \{x \in \mathbb{R}^n;\ \exists y \in \mathbb{R},\ (x, y) \in S\}$ be the projection on $\mathbb{R}^n$ of a set $S \subseteq \mathbb{R}^{n+1}$. If S is a semialgebraic set with coefficients in $\mathbb{Z}$ (respectively, $\mathbb{R}$), S' is a semi-algebraic set with coefficients in $\mathbb{Z}$ (respectively, $\mathbb{R}$).*

An important consequence of the Tarski–Seidenberg principle is the elimination of quantifiers: a set defined by a logical formula in the first-order theory of the reals is semialgebraic. These formulas are defined by induction as follows.

DEFINITION 2 (First-order formulas). P op 0 is a first-order formula for $P \in \mathbb{Z}[X_1, ..., X_n]$ and op $\in \{<, \leqslant, =, \geqslant, >\}$. If F_1 and F_2 are first-order formulas, $F_1 \vee F_2$, $F_1 \wedge F_2$, $\forall x \in \mathbb{R} F_1$, and $\exists x \in \mathbb{R} F_1$ are first-order formulas.

If there are n free variables $x_1, ..., x_n$ in a formula F, we write also $F(x_1, ..., x_n)$ instead of F. The subset of $\mathbb{R}^n$ defined by F is

$$S = \{(x_1, ..., x_n) \in \mathbb{R}^n;\ F(x_1, ..., x_n) \text{ is true}\}.$$

THEOREM 2 (Elimination of quantifiers). *The set $S \subseteq \mathbb{R}^n$ defined by a first-order formula $F(x_1, ..., x_n)$ is semialgebraic.*

Proof Sketch. Semialgebraic sets are clearly closed under boolean operations. The Tarski–Seidenberg principle amounts to the elimination of one existential quantifier. A universal quantifier can be replaced by an existential quantifier since $\neg \exists x \neg F$ is equivalent to $\forall x\, F$. ∎

3. SIMULATING REAL TURING MACHINES

3.1. *Presentation of the Model*

Blum, Shub, and Smale proposed a model of computation (the real Turing machine) that can be seen as a RAM machine in which a register can store an arbitrary real number with infinite precision. The constants occurring in the program executed by a real Turing machine (RTM in the sequel) are also arbitrary real numbers, and the basic operations ($+$, $-$, $\times$, $\div$, $\leqslant$) can be performed in constant time. We will not go into a formal description since this model is described in great detail in [4] (see also [11], where more general models are studied). It will be sufficient for us to know that a RTM is equipped with the usual instructions necessary to write programs, such a loops and if-then-else statements.

Throughout the paper, $\alpha_1, ..., \alpha_p$ are the nonzero constants occurring in a program.

A computation tree can be associated to a program as follows: each node of the tree is labeled by an instruction. A test node has two children which represent the two possible outcomes of the test. The other internal nodes have a single child. At each node $\mathscr{N}$, the content of a variable x is a rational function $Q_{\mathscr{N}, x}(u, \alpha)$ of the input $u \in \mathbb{R}^n$ and the constants $\alpha = (\alpha_1, ..., \alpha_p)$.

We shall in fact work in a modified BSS model in which the cost of an arithmetic operation is *not* constant.

DEFINITION 3. *If $x = Q(u, \alpha)$ after an arithmetic operation $x := y$ op z, the cost of this operation in the weak BSS model is the maximum of $d(Q)$ and the maximum size of the coefficients of Q (we assume that this rational fraction is irreducible).*

This model is weaker than the standard one since the class of polynomial-time computable functions is (presumably) smaller than in the full BSS model. For example, it is possible to output integers of exponential length in polynomial time with a program using only integer constants in the full model. This is clearly impossible in the weak model (even if real constants are allowed, as could be shown by generalizing Theorem 4 to integer-valued outputs).

The weak BSS model is interesting not only because of Theorems 4 and 15. It is perhaps a more realistic model of feasible analog computation than the full BSS model since it is not difficult to see that the standard notion of discrete polynomial-time computability is recovered when the constants $\alpha_1, ..., \alpha_p$ are rational (see Theorem 3). Note that the perhaps undesirable possibility to create integers of exponential length in polynomial time in the full model is a crucial tool in the recent proof that $\mathrm{P}_R \neq \mathrm{NC}_R$ [8]. This argument fails to separate weak polynomial time from NC_R.

The following easy characterization will be useful.

LEMMA 1. *A function is polynomial-time computable in the weak BSS model if and only if it is polynomial-time computable in the standard BSS model and the rational fractions $Q_{\mathcal{N}, x}$ have polynomial degree and coefficients of polynomial size.*

Proof. If F is computable in time $T(n)$ in the weak model, it is also computable in time $T(n)$ in the full model and the degrees and coefficient sizes of the rational fractions $Q_{\mathcal{N}, x}$ are bounded by $T(n)$.

If F is computable in time $T(n)$ in the full model with degrees and sizes bounded by $S(n)$, F is computable in the weak model in time $T(n)\, S(n)$. ∎

The weak BSS model includes as a special case a model in which divisions are forbidden and a variable can be multiplied only by a constant (and not another variable). This simple subcase is sufficient for dealing with the neural nets of Section 8.

LEMMA 2. *Let F be a function computable in polynomial-time in the standard BSS model without divisions by a program in which a variable is not multiplied by another variable. Then F is polynomial-time computable in the weak BSS model.*

Proof. The rational fractions $Q_{\mathcal{N}, x}$ are in fact polynomials since no division is performed. Let us show by induction that at time t, the content of a variable x is $P(\alpha, u)$, with $d(P) \leqslant t + 1$ and the coefficients of P bounded by 2^t.

The result clearly holds for $t = 0$. Assume that the result holds at time t. At time $t + 1$, it suffices to consider the case of a multiplication by a constant $x := \alpha_i y$ or an addition $x := y + z$. Let $y = P(\alpha, u)$ and $z = Q(\alpha, u)$. $d(\alpha_i P) = d(P) + 1$, and the coefficients do not change. $d(P + Q) \leqslant \max(d(P), d(Q))$, and the coefficients can be at most doubled. ∎

Let us now introduce some notations. P_R and P_W are the classes of real languages, i.e., subsets of

$$\mathbb{R}^\infty = \bigcup_{n=0}^{\infty} \mathbb{R}^n$$

that can be recognized in polynomial time in the BSS model and the weak BSS model, respectively. NP_R is the nondeterministic counterpart of P_R. More precisely, a language L is in NP_R if there is a language $A \in \mathrm{P}_R$ and a polynomial p such that for every n and every $u \in \mathbb{R}^n$,

$$u \in L \Leftrightarrow \exists x \in \mathbb{R}^{p(n)} (x, u) \in A.$$

One could also define a weak version NP_W of NP_R by taking A in P_W instead of P_R. However, it turns out that $\mathrm{NP}_W = \mathrm{NP}_R$ (see [10] and Section 7). Digital nondeterminism is also of interest. In this case, the guess u is taken in $\{0, 1\}^{p(n)}$ instead of $\mathbb{R}^{p(n)}$. This gives rise to the classes DNP_R and DNP_W.

3.2. *Boolean Inputs*

In the rest of this paper, we are mainly interested in computing boolean functions on $\{0, 1\}^*$, or, equivalently, recognizing boolean languages (subsets of $\{0, 1\}^*$). We thus address the following question: what can be gained from the availability of arbitrary real constants? (see Section 7 for a result of a different nature.) The boolean part $\mathrm{BP}(\mathscr{C})$ of a class $\mathscr{C}$ of real languages is the restriction of $\mathscr{C}$ to boolean inputs. More precisely,

$$\mathrm{BP}(\mathscr{C}) = \{L \cap \{0, 1\}^*; L \in \mathscr{C}\}.$$

For instance, $\mathrm{BP}(\mathrm{P}_W)$ is the class of boolean languages which can be recognized in polynomial-time in the weak BSS model. Note that in the BP (NP_R) case, the inputs are boolean but the guesses are real numbers.

We will sometimes consider machines which cannot use arbitrary real constants (see Section 7). If $\mathscr{C}$ is a class of languages recognized by resource-bounded real machines, $\mathscr{C}(E)$ denotes the class of languages recognized within the same resource constraints by real machines using only constants from the set $E \subseteq \mathbb{R}$. For example, $\mathrm{P}_R(\mathbb{R}) = \mathrm{P}_R$ and

$BP(P_W(0, 1))$ is the set of boolean languages recognized in weak polynomial time by machines using only the constants 0 and 1.

THEOREM 3. $BP(P_W(\mathbb{Q})) = P$.

Proof. If $L \in BP(P_W(\mathbb{Q}))$, it follows from Lemma 1 that $L \in P$ since during a computation, each register of the real Turing machine contains a rational number of polynomial size.

Clearly, $P \subseteq BP(P_W(\mathbb{Q}))$ since real machines are at least as powerful as Turing machines (in fact, $P \subseteq BP(P_W(0, 1))$). ▮

We are now in position to state our main result.

THEOREM 4. $BP(P_W) = P/poly$.

Proof. In order to show that $P/poly \subseteq BP(P_W)$, we encode an advice function in the digits a_i of a real constant $\alpha = \overline{0 \cdot a_1 a_2 \cdots a_i \cdots}$. Let us choose an encoding scheme such that infinitely many a_i's are equal to 0. The value of a_1 can be easily computed since $a_1 = 0 \Leftrightarrow \alpha < 1/2$. In order to compute a_2, we shift α to the left with the instruction $\alpha := 2\alpha - a_1$. Now a_2 can be computed like a_1. By iterating this process, it is possible to retrieve as many digits of c as desired. Since the advice are of polynomial size, the whole decoding process takes polynomial time. According to Lemma 2, it actually takes weak polynomial time.

We prove in Section 5 that $BP(P_W) \subseteq P/poly$, with the help of a technical result established in Section 4. ▮

COROLLARY 1. *If* $P_W = DNP_W$, *the polynomial hierarchy collapses at the second level.*

Proof. According to Theorem 4, $P_W = DNP_W \Rightarrow NP \subseteq P/poly$ since $NP \subseteq BP(DNP_W)$. If $NP \subseteq P/poly$, the polynomial hierarchy collapses at the second level [2]. ▮

As mentioned in the Introduction, it was recently shown in [10] that $P_W \neq NP_W = NP_R$ holds without any assumption.

The decoding argument of Theorem 4 also shows that all boolean languages can be recognized in weak exponential time.

THEOREM 5. $BP(EXP_W) = 2^{\{0,1\}^*}$.

Proof Sketch. The truth table of any boolean language can be encoded in the digits of a real constant, and it can be decoded in weak exponential time as in Theorem 4 (there are 2^n digits for inputs of size n). ▮

3.3. *Hardness and Feasibility Results*

In this paper the emphasis is on the solution of discrete problems with real "black boxes." This was surely not the point of view of Blum, Shub, and Smale when they proposed their model. They were much more interested in the solution of continuous problems. However, our point of view is not too restrictive for hardness results, since an algorithm solving a continuous problem must in particular provide the correct answer for discrete inputs. (Moreover, some techniques designed for dealing with boolean inputs and real constants can be adapted to the case of real inputs and boolean constants; see Section 7.) We can thus give evidence of hardness for problems in DNP_W such as the real traveling salesman problem of [4], the real knapsack problem or other continuous generalizations of NP-complete problems. More precisely, if one of these problems can be solved in polynomial time in the weak BSS model, $NP \subseteq P/poly$ according to Theorem 4, and the polynomial hierarchy collapses at the second level.

In our opinion, many (if not all) natural problems that can be solved in polynomial time in the standard BSS model can in fact be solved in polynomial time in the weak model. This is obviously true for very easy problems such as multiplication of polynomials or matrices. Let us give two other nontrivial examples.

The first problem consists of determining the sign of a real polynomial P at the root of another real polynomial Q. Two algorithms for this problem are described in [9, 18], and it is shown that the conditions of Lemma 1 are satisfied. Hence they run in polynomial time in the weak BSS model. Note that the algorithm of [9] is used in the same reference as a building block for computing the topology of a real algebraic plane curve.

Our second example is simply the resolution of systems of linear equations. This problem can obviously be solved in polynomial time in the BSS model by Gaussian elimination. For rational data, it is known that this procedure also runs in polynomial time in the bit model [21, Theorem 3.3]. It can be proven as an intermediate result that during the elimination procedure the coefficients of the system have a very special form; they are quotients of determinants extracted from the matrix of the original system. Note that as a polynomial function of the entries, the determinant of a matrix has polynomial (in fact, sublinear) degree, and coefficients of polynomial (in fact, constant) size. According to Lemma 1, this shows that Gaussian elimination runs in polynomial time in the weak model.

4. SMALL POINTS IN SEMIALGEBRAIC SETS

The proof of Theorem 4 relies on the following technical result.

THEOREM 6. *Let* $\mathcal{S} \subseteq \mathbb{R}^p$ *be defined by a system of inequalities*

$$P_i(x) > 0, \quad i = 1 \cdots N,$$

with $P_i \in \mathbb{Z}[X_1, ..., X_p]$. Let D be the maximum degree of the P_i's, and let S be the maximum size of their coefficients. If $\mathscr{S} \neq \emptyset$, there exists a rational point $x \in \mathscr{S}$ of size $s(x) \leqslant aSD^b$. The constants a and b depend only on p.

This result states that if $\mathscr{S} \neq \emptyset$ is defined by "small" polynomials, it contains a "small" rational point. Note that the number of inequations does not come into play in this bound.

Before proving Theorem 6, we present some useful results from computer algebra. These tools are necessary for the proof of Lemma 5, but the rest of this section can be read independently.

4.1. *Basic Tools*

The following bound is well known.

LEMMA 3 (Cauchy, 1829). *Let $P(x) = \sum_{i=0}^{D} a_i x^i$ $(a_D \neq 0)$ be a complex polynomial, and*

$$m = \inf\{|a_i|, i = 0 \cdots D, a_i \neq 0\}, \quad M = \sup\{|a_i|, i = 0 \cdots D\}.$$

If $\alpha \in \mathbb{C}$ (set of complex numbers), *$\alpha \neq 0$ is a root of P then*

$$\frac{m}{m+M} < |\alpha| < \frac{m+M}{m}.$$

The sign vector of a sequence of polynomials $P_1, ..., P_N \in \mathbb{R}[X]$ at a point $x \in \mathbb{R}$ is the vector $(\varepsilon(P_i(x)))_{1 \leqslant i \leqslant N} \in \{-1, 0, 1\}^N$, with, by definition, $\varepsilon(y) = 1$ if $y > 0$, 0 if $y = 0$, -1 if $y < 0$. Usually, a real root α of a polynomial $P \in \mathbb{R}[X]$ is coded by an interval $]a, b[$ containing α and no other root of P. A purely symbolic coding method was proposed in [7]: the roots of P are characterized by the sign vectors of the derivatives of P. More precisely, if $d(P) \leqslant D$, α is characterized by the sign vector

$$\varepsilon(\alpha) = (\varepsilon(P^{(i)}(\alpha))_{1 \leqslant i \leqslant D} \in \{-1, 0, 1\}^D.$$

According to Thom's lemma, a root is uniquely determined by its sign vector: if $P(\alpha) = P(\beta) = 0$ and $\varepsilon(\alpha) = \varepsilon(\beta)$, then $\alpha = \beta$. It is, moreover, possible to compare two roots α and β by comparing only their sign vectors. More formally, there exists a comparison function c_D: $\{-1, 0, 1\}^D \times \{-1, 0, 1\}^D \to \{-1, 0, 1\}$ independent of P such that $c_D(\varepsilon(\alpha), \varepsilon(\beta)) = -1$ if $\alpha < \beta$, 0 if $\alpha = \beta$, 1 if $\alpha > \beta$; c_D can be actually computed with a very simple algorithm (see [7] for more details).

Given $a \in \mathbb{R}^{D+1}$, let P_a be the polynomial $P_a(X) = \sum_{i=0}^{D} a_i X^i$, and $r(P_a)$ the number of distinct real roots of P_a. The following key result is a special case of the "quantifier elimination subproblem" of [18]. Similar bounds were also provided in [9].

LEMMA 4 (Renegar). *Let $N_1, .., N_M$ be positive integers, and $\varepsilon_1, ..., \varepsilon_M \in \{-1, 0, 1\}^N$ sign vectors. The set $S_{a,b} \subseteq \mathbb{R}^{N(D+1)} \times \mathbb{R}^{D+1}$ of coefficients $(a, b) = (a_1, ..., a_N, b) \in \mathbb{R}^{D+1} \times \mathbb{R}^{D+1} \times \cdots \times \mathbb{R}^{D+1}$ satisfying the following two properties is semi-algebraic*:

(i) $r(P_b) = N_1 + \cdots + N_M$;

(ii) *N_j is the number of distinct real zeros of P_b at which the sign vector of $(P_{a_i})_{1 \leqslant i \leqslant N}$ is ε_j for all $j = 1, ..., M$.*

Moreover, $S_{a,b}$ can be defined by polynomial (in)equalities of degree $(ND)^{O(1)}$ with coefficients of size $(ND)^{O(1)}$.

4.2. *Proof of Theorem 6*

Given a polynomial $P \in \mathbb{R}[X]$, we denote by α_ε its root of sign vector ε—if there is such a root. In this section, the following notation will be used: $]a, b[$ denotes the interval $]\min(a, b), \max(a, b)[$.

LEMMA 5. *For $D \geqslant 1$, let $S_{\varepsilon,\varepsilon'} \subseteq \mathbb{R}^{D+1} \times \mathbb{R}^{D+1}$ be the set of coefficients (a, b) such that*

$$P_b(x) > 0 \qquad \text{for} \quad x \in]\alpha_\varepsilon, \alpha_{\varepsilon'}[, \tag{1}$$

where α is the sequence of roots of P_a (if α_ε or $\alpha_{\varepsilon'}$ does not exist, we set $S_{\varepsilon,\varepsilon'} = \emptyset$). $S_{\varepsilon,\varepsilon'}$ is a semialgebraic set defined by polynomials of degree $D^{O(1)}$ whose coefficients are of size $D^{O(1)}$.

Proof. We can assume without loss of generality that $c_D(\varepsilon, \varepsilon') = -1$. This simplies that $\alpha_\varepsilon < \alpha_{\varepsilon'}$ whenever these roots exist. Lemma 4 will be applied several times to P_a, P_b, $P_a P_b$ and their derivatives. We call a semialgebraic set "small" if it is defined by polynomials of degree $D^{O(1)}$ whose coefficients are of size $D^{O(1)}$.

Let $B_{\varepsilon,\varepsilon'}$ be the set of coefficients $(a, b) \in \mathbb{R}^{2(D+1)}$ such that $P_b \neq 0$ on $]\alpha_\varepsilon, \alpha_{\varepsilon'}[$. In order to compare the roots of P_a and P_b, we consider the polynomial $P_a P_b$ (whose zeros are the roots of P_a and P_b). According to Lemma 4 applied to P_a, $P_a P_b$ and their derivatives, $B_{\varepsilon,\varepsilon'}$ is a small semialgebraic set, since $(a, b) \in B_{\varepsilon,\varepsilon'}$ if

1. $r(P_a P_b) = r \geqslant 2$;
2. the sign vectors of the sequence of $2D + 1$ polynomials

$$(((P_a P_b)^{(k)})_{1 \leqslant k \leqslant 2D}, P_a, (P_a^{(k)})_{1 \leqslant k \leqslant D})$$

at the zeros of $P_a P_b$ are

$$(\sigma_i, \alpha_i, \varepsilon_i) \in \{-1, 0, 1\}^{2D} \times \{-1, 0, 1\} \times \{-1, 0, 1\}^D$$

for $1 \leqslant i \leqslant r$.

3. $c_{2D}(\sigma_i, \sigma_j) = -1$ for all $i < j$, and there exists $l < r$ such that $\alpha_l = \alpha_{l+1} = 0$, $\varepsilon_l = \varepsilon$, and $\varepsilon_{l+1} = \varepsilon'$.

If $P_b \neq 0$ on $]\alpha_\varepsilon, \alpha_{\varepsilon'}[$, P_b is either positive or negative on this interval. More precisely, P_b will take the sign of its first nonzero derivative at α_ε. We thus define C_ε as the set of coefficients (a, b) such that the first nonzero derivative of P_b at α_ε is positive. According to Lemma 4 applied to P_a, P_b, and their derivatives, C_ε is a small semialgebraic set, since $(a, b) \in C_\varepsilon$ if

1. $r(P_a) = r \geqslant 1$;
2. the sign vectors of the sequence of $2D+1$ polynomials

$$((P_\alpha^{(i)})_{1 \leqslant i \leqslant D}, (P_b^{(i)})_{0 \leqslant i \leqslant D})$$

at the zeros of P_a are $(\varepsilon_1, \sigma_1), ..., (\varepsilon_r, \sigma_r)$;

3. there exists j such that $\varepsilon_j = \varepsilon$ and the first nonzero component of σ_j is positive.

The claim now follows from the relation $S_{\varepsilon, \varepsilon'} = B_{\varepsilon, \varepsilon'} \cap C_\varepsilon$. ∎

LEMMA 6. *Given $a \in \mathbb{R}^{N(D+1)}$, let $(P_{a_i})_{1 \leqslant i \leqslant N}$ be the sequence of polynomials such that $P_{a_i} = \sum_{j=0}^{D} a_{ij} X^j$. Let*

$$S_a = \{x \in \mathbb{R},\ P_{a_1}(x) > 0 \wedge P_{a_2}(x) > 0 \wedge \cdots \wedge P_{a_N}(x) > 0\}.$$

The set of coefficients

$$\mathscr{C} = \{a \in \mathbb{R}^{N(D+1)},\ \mathscr{S}_a \neq \varnothing\}$$

is a semi algebraic subset of $\mathbb{R}^{N(D+1)}$ defined by polynomials of degree $D^{O(1)}$ whose coefficients are of size $D^{O(1)}$. No more than $3(D+1)$ of the $N(D+1)$ variables a_{ij}'s can occur in a given polynomial.

Proof. If $\mathscr{S}_a \neq \varnothing$ for a given $a \in \mathbb{R}^{N(D+1)}$, there exists a maximal nonempty interval $]\alpha, \beta[\subseteq \mathscr{S}_a$. If α and β are finite, they are roots of two polynomials P_{a_i} and P_{a_j} (possibly, $i = j$). Hence the set $\mathscr{C}_1$ of vectors $(a, b) \in \mathscr{C}$ corresponding to the case where both α and β are finite satisfies:

$$\mathscr{C}_1 = \bigcup_{i, j, \varepsilon, \varepsilon'} \bigcap_{l} \mathscr{C}_{ij\varepsilon\varepsilon' l},$$

where

$$\mathscr{C}_{ij\varepsilon\varepsilon' l} = \{a \in \mathbb{R}^{N(D+1)},\ P_{a_l} > 0 \text{ on }]\alpha_{ij\varepsilon}, \alpha_{ij\varepsilon'}[\}$$

and $\alpha_{ij\eta}$ is the root of $P_{a_i} P_{a_j}$ of sign vector η. The conclusion of the lemma holds for each $\mathscr{C}_{ij\varepsilon\varepsilon' l}$ according to Lemma 5; hence it holds also for $\mathscr{C}_1$. The cases where α or β are infinite can be handled with a straightforward adaptation of Lemma 5. ∎

Proof of Theorem 6. By induction on p. For $p = 1$, as was noticed in the proof of Lemma 6, there exists a maximal nonempty interval $]\alpha, \beta[\subseteq \mathscr{S}$ whose endpoints, when finite, are roots of two polynomials P_i and P_j. It follows from Lemma 3 that α and β cannot be too large (if they are finite). $\beta - \alpha$ cannot be too small since it is the root of a polynomial of degree $D^{O(1)}$ with integer coefficients of size $O(SD^{O(1)})$ [15]. Hence there are small rational points in $]\alpha, \beta[$.

Assume now that the result holds in dimension $p-1$. In order to prove that it holds for $\mathscr{S} \subseteq \mathbb{R}^p = \mathbb{R}^{p-1} \times \mathbb{R}$, $S \neq \varnothing$, we consider the projection $\mathscr{S}'$ of $\mathscr{S}$ on $\mathbb{R}^{p-1}$. It will be shown that $\mathscr{S}'$ is semialgebraic (this is just the Tarski–Seidenberg principle) and described by small polynomials. Hence S' contains a small point x_0 by induction hypothesis. The polynomials $P_i(x_0, x_p)$ have small coefficients; hence there exists a small x_p such that $(x_0, x_p) \in \mathscr{S}$. We now fill in the missing details.

Let us consider each P_i as a polynomial in x_p only; one can write

$$P_i(x) = \sum_{j=0}^{D} a_{ij}(x') x_p^j$$

with $x' = (x_1, x_2, ..., x_{p-1})$. In the notations of Lemma 6, $x' \in \mathscr{S}'$ if and only if $a(x') = (a_{ij}(x'))_{i,j} \in \mathscr{C}$. Let c and d be the constants such that the degrees of the polynomials describing $\mathscr{C}$ and the size of their coefficients are smaller than cD^d. Let P be a polynomial occurring in the description of S': P is the composition of two maps $Q: \mathbb{R}^{p-1} \to \mathbb{R}^k$ and $R: \mathbb{R}^k \to \mathbb{R}$, where $k \leqslant 3(D+1)$ by Lemma 6. The components of Q are selected among the a_{ij}'s; hence they are polynomials of degree at most D and have coefficients of size at most S. By construction, R is a polynomial whose degree and coefficient size is bounded by cD^d. Therefore P is of degree at most cD^{d+1}, and the size of its coefficients is bounded by $c'SD^{d'}$ for some constants c' and d'. The decomposition of S' in basic semialgebraic sets is of the form

$$S' = \bigcup_{i=1}^{m} S_i'$$

with

$$S_i' = \{x' \in \mathbb{R}^{p-1};\ Q_{i1} > 0 \wedge \cdots \wedge Q_{ij_i} > 0 \wedge R_{i1} = 0 \wedge \cdots \wedge R_{ik_i} = 0\}.$$

$\mathscr{S}'$ is open since $\mathscr{S}$ is open; therefore $k_i = 0$ for at least one of the $\mathscr{S}_i'$'s, say, $\mathscr{S}_1'$. By induction hypothesis, there exist constants a' and b' and a point $x_0 \in \mathscr{S}_1'$ of size

$$s(x_0) \leqslant a'(c'SD^{d'})(cD^{d+1})^{b'};$$

there are clearly constant a and b such that $s(x_0) \leqslant aSD^b$. A similar bound can be obtained for x_p by substituting x_0 to x' in the polynomials $P_i(x', x_p)$, clearing the denominators and applying Theorem 6 in dimension 1. ∎

5. PROOF OF THE MAIN RESULT

In this section, we complete the proof of Theorem 4. From now on, it is assumed without loss of generality that all comparisons are of the form "$x > 0$," where x is a variable. The proof will first be made under the following hypothesis, and then in the general case:

(H) Whenever a test $x > 0$ is performed, x is not equal to zero.

Two solutions will be provided for the general case. In this section, we shall see that (H) can be assumed without loss of generality. It is shown in Section 6 that equality cases can be handled with symbolic computation techniques. Note that (H) holds if the α_i's are algebraically independent.

THEOREM 7. *If a language $L \subseteq \{0, 1\}^*$ can be recognized in polynomial time in the weak BSS model under hypothesis* (H), *then* $L \in \mathrm{P/poly}$.

Proof. The main idea is to replace the constants $\alpha_1, ..., \alpha_p$ by rational approximations $\overline{\alpha_1}, ..., \overline{\alpha_p}$. In accordance with the definition of advice functions, these constants must not fully depend on the input u, but only on its length $n = |u|$. The program is then run with these new constants: at time t, the content of the variable x is now $\bar{x}(t)$ instead of $x(t)$. Two points need to be checked:

1. The outcome of the computation is the same.
2. The computation remains polynomial-time.

The first condition will be satisfied if for any variable x and time t, $x(t) < 0 \Rightarrow \bar{x}(t) < 0$ and $x(t) > 0 \Rightarrow \bar{x}(t) > 0$. As a matter of fact, these requirements ensure that the results of the tests (including the final test on R) will be the same in both programs. Let $T(n) = O(n^l)$ be the worst-case running time for inputs of length n. According to Lemma 1, we obtain a system of polynomial equations on $\bar{\alpha} = (\overline{\alpha_1}, ..., \overline{\alpha_p})$ of the type

$$\mathscr{E} = \{ P_{t,u}(\bar{\alpha}) > 0 \}_{t \leqslant T(n),\ |u| = n}.$$

An equation $P_{t,u} > 0$ occurs in $\mathscr{E}$ if the construction executed at time t on input u is a test on a variable x, and in this case

$$P_{t,u} = \mathrm{sign}(x(t)) \cdot R_{\mathscr{N},x} \cdot S_{\mathscr{N},x},$$

$R_{\mathscr{N},x}$ and $S_{\mathscr{N},x}$ being the polynomials such that $Q_{\mathscr{N},x} = R_{\mathscr{N},x}/S_{\mathscr{N},x}$ (the rational fractions $Q_{\mathscr{N},x}$ are defined in Section 3). $\mathscr{E}$ is satisfiable system, since α is obviously a solution. Theorem 6 and Lemma 1 will now be invoked to show that α can be replaced by a rational approximation which needs not be too accurate. This will ensure the second condition.

According to Lemma 1, the $P_{t,u}$'s are of polynomial degree and their coefficients of polynomial size. It follows from Theorem 6 that $\mathscr{E}$ has a rational solution $\bar{\alpha}$ of polynomial size. The approximations $\bar{x}(t)$ can thus be computed in polynomial time. ∎

A bound similar to Theorem 6 where the maximum degree D is replaced by the sum of the degrees can be found in [19]. Such a result cannot be used here since $\mathscr{E}$ can contain an exponential number of inequations. It is therefore crucial that this number does not come into play in the bound of Theorem 6 (or only very mildly).

We now show how to get rid of equality cases in the general case; this shows that (H) is in fact not restrictive. This solution is based on the following obvious remark: if a test "$x > 0$" is performed during the execution of a RTM program $\mathscr{P}$, and if $x \neq 0$, there exists $\varepsilon > 0$ such that either $x > \varepsilon$ or $x < -\varepsilon$. For a fixed n, only a finite number of tests are performed during the computations on all inputs of length n; hence we can choose the same ε for all these tests. For each n, we define a new system $\mathscr{E}''$ of inequations on $\overline{\alpha_1}, ..., \overline{\alpha_p}$ and on a new variable $\bar{\varepsilon}$. $\mathscr{E}''$ is obtained from $\mathscr{E}$ by the following operations:

1. add the inequation $\bar{\varepsilon} > 0$;
2. replace the inequation associated to a test "$x > 0$" by the inequation associated to the test:
   (a) $x - \bar{\varepsilon} > 0$ if it turns out that $x > 0$;
   (b) $x - \bar{\varepsilon} < 0$ if it turns out that $x \leqslant 0$.

$\mathscr{E}''$ is satisfiable since $(\alpha_1, ..., \alpha_p, \varepsilon)$ is a solution; hence there exists a "small" solution $(\overline{\alpha_1}, ..., \overline{\alpha_p}, \bar{\varepsilon})$ according to Theorem 6. The program obtained by replacing the α_i's by the $\bar{\alpha}_i$'s and the tests "$x > 0$" by "$x > \bar{\varepsilon}$" runs in polynomial time and is equivalent to $\mathscr{P}$.

The symbolic solution to the problem of equality cases is described in detail in the next section. For each test "$x > 0$," we test whether x is equal to 0 with the algorithm of Theorem 8. If it turns out that $x = 0$, then of course we know that the answer to the original test is negative. If it turns out that $x \neq 0$, then we can use small relational constants $\overline{\alpha_i}$ as above.

6. ORDER-FREE MODELS

In this section, we consider computation models for which the order relation is not available. Therefore the only authorized tests are equality tests. We first give a simulation result for the weak BSS model and extend it to the full (order-free) BSS model by probabilistic methods. The corresponding complexity classes are denoted by the "$=$" superscript. As in Section 3.2, we consider only boolean inputs.

In order to perform a test $x \stackrel{?}{=} 0$, we will explicitly compute the polynomials R and S such that $x = R(\alpha)/S(\alpha)$. This can be done in polynomial time according to Lemma 1

(recall that a polynomial of degree d in p variables has only $O(d^p)$ monomials). We need to determine whether R belongs to the ideal $\mathscr{I} = \{P \in \mathbb{Z}[X], P(\alpha) = 0\}$. This problem can be solved by symbolic computation techniques because the ideals of $\mathbb{Z}[X]$ are finitely generated, i.e.,

$$\mathscr{I} = \left\{ \sum_{i=1}^{m} Q_i G_i, Q_1, ..., Q_m \in \mathbb{Z}[X] \right\}$$

for some finite system of generators

$$\mathscr{G} = (G_1, ..., G_m) \in \mathbb{Z}[X]^m.$$

This follows from Hilbert's theorem on noetherian rings (see, e.g., [14, Chap. 6]). Hence we just have to determine if R belongs to the ideal generated by $\mathscr{G}$. One could use Gröbner basis techniques to solve this problem, but this is in fact not necessary. Bruno Poizat (personal communication) noticed that the primitive element theorem [14] suffices. The argument is as follows.

Let k be the transcendence degree over $\mathbb{Q}$ of the field $K = \mathbb{Q}[\alpha_1, ..., \alpha_p]$. We can assume without loss of generality that $\alpha_1, ..., \alpha_k$ is a transcendence base of K. The other constants $\alpha_{k+1}, ..., \alpha_p$ are algebraic over $\mathbb{Q}[\alpha_1, ..., \alpha_k]$. It follows from the primitive element theorem that

$$K = \mathbb{Q}[\alpha_1, ..., \alpha_k][\alpha_{k+1}, ..., \alpha_p] = \mathbb{Q}[\alpha_1, ..., \alpha_k][\beta],$$

where the primitive element $\beta \in \mathbb{R}$ is algebraic over $\mathbb{Q}[\alpha_1, ..., \alpha_k]$. Hence there are rational fractions Q_j such that

$$\alpha_j = Q_j(\alpha_1, ..., \alpha_k, \beta), \quad j = k+1, ..., p, \tag{2}$$

and the content of register x is actually a rational fraction $x = R(\alpha_1, ..., \alpha_k, \beta)$. This property yields a very convenient way of testing whether x is equal to zero. We just have to make sure that as a polynomial in β, the numerator P_1 of the fraction $R = P_1/P_2$ is a multiple of the minimum polynomial M of β over $\mathbb{Q}[\alpha_1, ..., \alpha_k]$. Checking this condition is not difficult: perform the division of P_1 by M, and since $\alpha_1, ..., \alpha_k$ are algebraically independent, the remainder $R = P_1 = \mathrm{mod}\ M$ is equal to zero if its coefficients (which are in $\mathbb{Q}[\alpha_1, ..., \alpha_k]$) are indentically equal to zero.

We can assume that M and the Q_j's are "hardwired" in the program, since they are independent of the input. The whole computation therefore takes polynomial time. We have proved the following result.

THEOREM 8. $\mathrm{BP}(\mathrm{P}_W^=) = \mathrm{P}$.

Note that the symbolic technique developed in this section is more powerful than the "numerical" technique used in the previous section to deal with equality cases. Indeed, this latter technique would only show that $\mathrm{BP}(\mathrm{P}_W^=) \subseteq \mathrm{P/poly}$, instead of the stronger result $\mathrm{BP}(\mathrm{P}_W^=) = \mathrm{P}$. In the full BSS model, we are only able to prove the following weaker result.

THEOREM 9. $\mathrm{BP}(\mathrm{P}_R^=) \subseteq \mathrm{BPP}$.

Recall that BPP is the class of (boolean) languages that can be recognized in polynomial time by a probabilistic Turing machine with a bounded probability of error (see, e.g., [1] for more details). Before moving to the proof of Theorem 9, we present two lemmas which are often used in the analysis of randomized algorithms (see [12] for a proof of the first one).

LEMMA 7 (Schwartz). *Let $P(x_1, ..., x_p)$ be a polynomial of degree d. Choose the x_i's independently from the uniform distribution on $\{0, 1, ..., N-1\}$. If $P \not\equiv 0$, then*

$$\Pr\{P(x_1, ..., x_p) = 0\} \leqslant dp/N.$$

LEMMA 8. *Let $x \neq 0$ be an integer smaller than 2^{2^n}. For any constant $c > 1$ there is a constant $d > 0$ such that the number of integers smaller than 2^{cn} not dividing x is at least $d2^{cn}/n$.*

Proof. By the law of prime numbers there are at least $\Omega(2^{cn})$ primes smaller than 2^{cn}. However, x cannot have more than 2^n prime divisors. ∎

An application of this lemma in a very similar context can be found in [20].

COROLLARY 2. *Let $x \neq 0$ be an integer smaller than 2^{2^n}. Choose $m_1, ..., m_v$ independently from the uniform distribution on $\{0, 1, ..., 2^{cn}-1\}$, for some constant $c > 1$. There is a constant $C > 0$ such that for every $\varepsilon > 0$ and any $v > Cn \log(1/\varepsilon)$,*

$$\Pr\{x \bmod m_1 = x \bmod m_2 = \cdots = x \bmod m_v = 0\} \leqslant \varepsilon.$$

Proof. By Lemma 8, the probability to be bounded is smaller than $(1 - d/n)^v$. ∎

Proof of Theorem 9. The algorithm of Theorem 8 no longer runs in polynomial time, since rational fractions of exponential degree can be created in polynomial time (by iterated multiplication). As a first modification, we can compute the pairs ($P_1 \bmod M$, $P_2 \bmod M$) instead of computing the rational fractions $R = P_1/P_2$ (P_1, P_2 and M are still viewed as elements of $\mathbb{Q}[\alpha_1, ..., \alpha_k][X]$). Hence we now work with polynomials of bounded degree in β, but still face an exponential growth problem for their coefficients. These coefficients can be computed with a polynomial number of arithmetic operations in $\mathbb{Q}[\alpha_1, ..., \alpha_k]$. In order to decide if such a polynomial $P(\alpha_1, ..., \alpha_k)$ is equal to zero, we will not compute its coefficients explicitly. Instead, we replace $\alpha_1, ..., \alpha_k$ by randomly chosen integers $n_1, ..., n_k$ and perform the same sequence of arithmetic operations. If it turns out that $P(n_1, ... n_k) \neq 0$, then certainly $P \not\equiv 0$. If it turns out

that $P(n_1, ..., n_k) = 0$, then $P \equiv 0$ with high probability according to Lemma 7 (detailed analysis below). Unfortunately, this new sequence of arithmetic operations cannot be performed in polynomial time either, since we could create integers of exponential size. However, this problem can be solved with high probability by performing the operations modulo v randomly drawn integers $m_1, ..., m_v$ (Corollary 2).

Let us now show that in order to have a probability of error smaller than, say, 1/4, the running time of this probabilistic algorithm is indeed polynomial. Since the original program runs in polynomial time, it suffices to have a probability of error smaller than $1/n^{O(1)}$ for each test. For any given test, the probability of error is bounded by the sum of errors for the two steps:

1. application of Schwarz's lemma,
2. application of Corollary 2.

Since the polynomials in $\alpha_1, ..., \alpha_k$ occurring in the computations of the pairs ($P_1 \bmod M$, $P_2 \bmod M$) are obtained by a polynomial number of arithmetic operations, their degrees are smaller than $2^{n^{O(1)}}$. It follows from Schwarz's lemma that the randomly drawn integral points $(n_1, ..., n_k)$ at which these polynomials are evaluated can be taken of polynomial size.

It is also clear from Corollary 2 that in order to achieve an error probability smaller than $1/n^{O(1)}$, drawing a polynomial number of integers m_i of polynomial size is sufficient.

Let us now summarize the overall algorithm. It is easily verified that the same random integers $n_1, ..., n_k, m_1, ..., m_v$ can be used for all tests. Hence the first step of our algorithm will be:

1. On an input $u \in \{0, 1\}^n$, choose at random k integers $n_1, ..., n_k$ in $\{0, 1, ..., 2^{n^{O(1)}} - 1\}$ and $v = n^{O(1)}$ integers $m_1, ..., m_v$ in $\{0, 1, ..., 2^{n^{O(1)}} - 1\}$.

Each register or constant x of the real Turing machine will be represented by a sequence of rational fractions $(R_i^x)_{1 \leqslant i \leqslant v}$, where $R_i^x = P_i^x / Q_i^x$ and P_i^x, $Q_i^x \in \mathbb{Q}[X]$. These fractions are initialized as follows:

2. $R_i^{\alpha_j} = n_j \mod m_i$ for $1 \leqslant j \leqslant k$; $R_i^{\alpha_j}(X) = Q_j(n_1, ..., n_k, X)$ for $k+1 \leqslant j \leqslant p$. The coefficients of $R_i^{\alpha_j}$ are to be computed modulo m_i.

3. An operation $x := y$ op z is simulated by the following sequence of operations:

$$R_i^x := (R_i^y \text{ op } R_i^z) \bmod M(n_1, ..., n_k, X)$$

for $i = 1, ..., v$; the mod operator specifies that the numerator and denominator of R_i^x are to be computed modulo M. Moreover, the coefficients of R_i^x are to be computed modulo m_i.

4. For a test $x \stackrel{?}{=} 0$, a positive answer is provided if and only if $P_i^x = 0$ for every $i = 1, ..., v$.

At step 2, we assume without loss of generality that $\deg_X(Q_j) < \deg_X(M)$. This explains why, in contrast to step 3, we do not have to compute a remainder modulo M. ∎

7. A SEPARATION THEOREM

In this section, we show that the real polynomial hierarchy PH_R is strictly included in EXP_W, the class of real languages recognized in weak exponential time by real Turing machines. This generalizes considerably the large inclusion $\mathrm{NP}_R \subseteq \mathrm{EXP}_W$ which was established in [10]. The proof answers a question of [11]: is there a boolean language which is not in $\mathrm{BP}(\mathrm{P}_R)$? In fact, we show that for every k, some boolean language are not in $\mathrm{BP}(\Sigma_R^k)$, where Σ_R^k is the kth level of hierarchy (these languages are *a fortiori* not in $\mathrm{BP}(\mathrm{P}_R)$).

DEFINITION 4. A real language $L \subseteq \mathbb{R}^\infty$ is in Σ_R^k if there exists a set $A \in \mathrm{P}_R$ and polynomial functions $p_1, ..., p_k$ such that for any $u \in \mathbb{R}^n$, the following condition holds:

$$u \in L \Leftrightarrow Q_1 x_1 \in \mathbb{R}^{p_1(n)} \cdots Q_k x_k \in \mathbb{R}^{p_k(n)} (x_1, ..., x_k, u) \in A. \tag{3}$$

In this equation, the quantifiers $Q_i \in \{\exists, \forall\}$ alternate, starting with $Q_1 = \exists$. The class Π_R^k is defined by the same condition except that the first quantifier is universal ($Q_1 = \forall$). By definition,

$$\mathrm{PH}_R = \bigcup_{k=0}^{+\infty} \Sigma_R^k = \bigcup_{k=0}^{+\infty} \Pi_R^k.$$

One can also define a weak polynomial hierarchy in the obvious way by replacing the condition $A \in \mathrm{P}_R$ by $A \in \mathrm{P}_W$.

We need two results from real algebraic geometry. The following bound on the number of satisfiable sign conditions for a set of multivariate polynomials is due to Milnor [17]; see also [18] for an "elementary" proof.

THEOREM 10 (Milnor). *Let $\{g_1, ..., g_m\}$ be a set of of m polynomials in n variables. A sign condition $\varepsilon \in \{-1, 0, 1\}^m$ is said to be consistent if there exists some $x \in \mathbb{R}^n$ such that ε_i is the sign of $g_i(x)$, for every $i = 1, ..., m$. If the g_i's are of degree at most d, the number of consistent sign conditions is $(md)^{O(n)}$.*

We also need the following result of Renegar on quantifier elimination in the real number model [18].

THEOREM 11 (Renegar). *Let F be a formula of the first-order theory of the reals with k quantifier alternations: F is of the form*

$$Q_1 x_1 \in \mathbb{R}^{n_1} \cdots Q_k x_k \in \mathbb{R}^{n_k} G(x_1, ..., x_k, u),$$

where G is a quantifier-free formula, and the free variable u is in $\mathbb{R}^l$. Let m be a bound on the number of polynomials occurring in F, and d a bound on their degrees. Then $F(u)$ is equivalent to a quantifier-free formula $H(u)$. The number of polynomials occurring in H is at most $(md)^{2^{O(k)}lN}$, and their degrees at most $(md)^{2^{O(k)}N}$, where $N=\prod_{i=1}^{k} n_i$. Formula H can be constructed by an arithmetic circuit of size $(md)^{2^{O(k)}lN}$ and depth polynomial in 2^k, l, N, m, and d.

We begin with a few general remarks.

LEMMA 9. *Assume that $L\subseteq\mathbb{R}^{\infty}$ can be recognized in polynomial time by a real Turing machine $\mathscr{M}$. Then L can be recognized by a family $(C_n)_{n\in\mathbb{N}}$ of polynomial-size arithmetic circuits over the reals. Moreover, C_n uses the same real constants as $\mathscr{M}$, and C_n can be constructed in polynomial time by an ordinary Turing machine.*

The proof of this result will be omitted since it is really the same as in the boolean case (i.e., simulation of Turing machines by boolean circuits [1]). In fact, Lemma 9 can be generalized to models of computation over arbitrary structures (see [11]).

Now, let L be a language in Σ_R^k (or Π_R^k) and let $\mathscr{M}$ be a polynomial time real Turing machine which recognizes the language A of (3). It follows immediately from Lemma 9 that there exists a sequence $(C_n)_{n>0}$ of arithmetic circuits over the reals satisfying the following properties:

- The output of C_n on any input $(x_1, ..., x_k, x)\in \prod_{i=1}^{k}\mathbb{R}^{p_i(n)}\times\mathbb{R}^n$ and the output of $\mathscr{M}$ on input x and "guesses" $x_1, ..., x_k$ are identical.
- The size s_n of C_n is polynomially bounded.
- The real constants $\alpha=(\alpha_1, ..., \alpha_p)$ of C_n and $\mathscr{M}$ are identical.

Note that for every $x\in\mathbb{R}^n$, the property

$$Q_1 x_1\in\mathbb{R}^{p_1(n)}\cdots Q_k x_k\in\mathbb{R}^{p_k(n)} C_n(x_1, ..., x_k, x)=1$$

is equivalent to a first-order *formula* of the form

$$Q_1 x_1\in\mathbb{R}^{p_1(n)}\cdots Q_k x_k\in\mathbb{R}^{p_k(n)}\ \exists y\in\mathbb{R}^{h(n)} F_n(x_1, ..., x_k, y, x). \tag{4}$$

The quantifier-free formula F_n is of size $O(s_n)$ and $h(n)\leqslant s_n$. This equivalent formula is simply obtained by adding new quantified variables y_i representing the outputs of all internal gates of C_n. This transformation can also be performed in arbitrary structures [11]. The following generalization of the $\mathrm{NP}_R=\mathrm{NP}_W$ result of [10] now follows.

THEOREM 12. *For every $k\geqslant 1$, $\Sigma_R^k=\Sigma_W^k$ and $\Pi_R^k=\Pi_W^k$.*

Proof. By definition, $\Sigma_W^k\subseteq\Sigma_R^k$ and $\Pi_W^k\subseteq\Pi_R^k$. Conversely, let L be a language in Σ_R^k such that the innermost quantifier Q_k is existential. For every $x\in\mathbb{R}^n$, $x\in L$ if and only if (4) holds. This shows that L is in Σ_W^k since the two blocks of quantifiers $Q_k x_k$ and $\exists y$ can be concatenated, and deciding $F_n(x_1, ..., x_k, y, x)$ given $x_1, ..., x_k$, y and x can be done in weak polynomial time. The same argument applies to $L\in\Pi_R^k$.

Assume now that Q_k is universal. The complement $\bar{L}$ of $L\in\Sigma_R^k$ is in Π_R^k and by the argument above, $\bar{L}\in\Pi_W^k$. Therefore $L\in\Sigma_W^k$. This argument also applies to $L\in\Pi_R^k$. ∎

We are now ready to prove the main result of this section.

THEOREM 13. PH_R *is strictly included in* EXP_W.

We need two intermediate results.

LEMMA 10. $\mathrm{PH}_R\subseteq\mathrm{EXP}_W$.

Proof. In order to decide whether $x\in\mathbb{R}^n$ belongs to a language L of Σ_R^k or Π_R^k, construct formula (4) in (weak) polynomial time. It follows from Theorem 11 that given an input $x\in\mathbb{R}^n$, formula (4) can be decided in (full) parallel polynomial time. It can be easily seen [10, Lemma 9] that full parallel polynomial time is included in weak exponential time. ∎

LEMMA 11. *For every $k>0$, Σ_R^k is strictly included in EXP_W.*

Proof. It will be sufficient to show that $\mathrm{BP}(\Sigma_R^k)$ is strictly included in $\mathrm{BP}(\mathrm{EXP}_W)$ since we already know that $\Sigma_R^k\subseteq\mathrm{EXP}_W$. The boolean part of EXP_W is $2^{\{0,1\}^*}$; i.e., every boolean language is in $\mathrm{BP}(\mathrm{EXP}_W)$. We shall show by a counting argument that some boolean languages are not in $\mathrm{BP}(\Sigma_R^k)$.

Let $\mathscr{L}$ be a language of Σ_R^k, and let $(C_n^{\alpha})_{n>0}$ be the family of arithmetic circuits constructed above. We use the notation C_n^{α} instead of C_n to emphasize the role of real constants. If the constants α are replaced by new constants β, we obtain a new circuit C_n^{β}. The two circuits C_n^{α} and C_n^{β} recognize different sets in general. However, when the input x is boolean and $x_1, ..., x_k$ are interpreted as real guesses, we shall see that it is possible to bound the number of "truly different" real constants. To make this precise, we say that $\alpha\equiv\beta$ if for every input $x\in\{0,1\}^n$,

$$Q_1 x_1\in\mathbb{R}^{p_1(n)}\cdots Q_k x_k\in\mathbb{R}^{p_k(n)} C_n^{\alpha}(x_1, ..., x_k, x)=1$$
$$\Leftrightarrow Q_1 x_1\in\mathbb{R}^{p_1(n)}\cdots Q_k x_k\in\mathbb{R}^{p_k(n)} C_n^{\beta}(x_1, ..., x_k, x)=1.$$

We now show that there is a constant a (dependent on k) such that the number E of equivalences classes of $\equiv$ satisfies

$$E\leqslant 2^{ap(n+p)\, s_n^{k+1}\log s_n}. \tag{5}$$

Let us apply Theorem 11 to formula (4), with $N\leqslant s_n^{k+1}$, $m\leqslant s_n$, $l=n+p$, and $d=2$. We obtain an equivalent quantifier-free formula $H_n^{\alpha}(x)$ in $n+p$ variables. Let $g_1(x,\alpha), ..., g_M(x,\alpha)$ be the set of polynomials occurring

in H. The degree of g_i is bounded by $s_n^{O(s_n^{k+1})}$ and $M \leqslant s_n^{O((n+p)\, s_n^{k+1})}$. Consider the $2^n M$ polynomials $p_{i,x}(\alpha)$ of the form $\alpha \mapsto g_i(x, \alpha)$, where $x \in \{0,1\}^n$ and $i \in \{1, ..., M\}$. The number of equivalences classes of $\equiv$ is bounded by the number of consistent sign conditions for the $p_{i,x}$. According to Theorem 10, this number is in turn bounded by

$$(2^n s_n^{O((n+p)\, s_n^{k+1})} s_n^{O(s_n^{k+1})})^p.$$

This completes the proof of (5).

Let $B_{s,p}(n)$ be the set of boolean functions on $\{0,1\}^n$ which can be computed by a quantified circuit of size s with p real constants and k quantifier alternations. Since the number of circuits of size S is singly exponential, it follows from (5) that

$$\# B_{s,p}(n) \leqslant 2^{O(p(n+p)\, s^{k+2} \log s)}.$$

Now, set for instance $p(n) = n$ and $s_n = n^{\log n}$. It is easily verified that $\# B_{s_n, p(n)}(n) < 2^{2^n}$ for some large enough n_0 and every $n > n_0$. Hence for $n > n_0$ there always exists a boolean function $L_n \subseteq \{0,1\}^n$ which is not in $B_{s_n, p(n)}(n)$. Consider the language $L = \bigcup_{n > n_0} L_n$. This language is not in $\mathrm{BP}(\Sigma_R^k)$ since for every $L' \in \mathrm{BP}(\Sigma_R^k)$, $L' \cap \{0,1\}^n$ is in $B_{s_n, p(n)}(n)$ for n large enough. ▮

Proof of Theorem 13. For $x \in \{0,1\}^*$, let $\bar{x}$ be the rank of x in the lexicographical ordering on $\{0,1\}^*$. We define a language $L \subseteq \mathbb{R}^\infty$ as follows: an input of the form $(y, x_1, ..., x_n)$ with $y \in [0,1]$ and $(x_1, ..., x_n) \in \{0,1\}^n$ is accepted if bit number $\bar{x}$ of the binary expansion of y is equal to 1. All other inputs are rejected. by the familiar decoding algorithm, $L \in \mathrm{EXP}_W$. This language has the property that for any boolean language $K \subseteq \{0,1\}^*$, there exists $y \in [0,1]$ such that the real language $L_y = \{x \in \mathbb{R}^\infty; (x, y) \in L\}$ satisfies $K = L_y$ (hence y plays the role of an arbitrary real constant).

Assume now that $L \in \mathrm{PH}_R$. Then $L \in \Sigma_R^k$ for some k. For any $y \in [0,1]$, L_y is also in Σ_R^k. Hence $\mathrm{BP}(\Sigma_R^k)$ contains all boolean languages; this is in contradiction with the proof of Lemma 11. ▮

The proof of Lemma 11 relies on the somewhat questionable possibility of using arbitrary real constants in the program of a real machine. In fact, Theorem 13 holds even without real constants.

THEOREM 14. *Let $E \subseteq \mathbb{R}$ be an arbitrary set containing the constant* 0 *and* 1. *Then* $\mathrm{PH}_R(E)$ *is strictly included in* $\mathrm{EXP}_W(E)$.

Proof. The inclusion $\mathrm{PH}_R(E) \subseteq \mathrm{EXP}_W(E)$ follows from the same argument as in the $E = \mathbb{R}$ case—and from the fact that in Renegar's quantifier elimination algorithm, it is not necessary to use any real constant that does not appear in the original formula.

The real language L defined in the proof of Theorem 13 is in $\mathrm{EXP}_W(0, 1)$ and was shown to be outside PH_R. It is thus in $\mathrm{EXP}_W(E) \backslash \mathrm{PH}_R(E)$ for any set of constants E. ▮

8. APPLICATION TO ARTIFICIAL NEURAL NETWORKS

8.1 *Recurrent Networks*

We shall use the model of analog computation with recurrent neural nets proposed in [22]. Let us first recall briefly the main features of this model. The dynamics of a network of n interconnected neurons $x_1, ..., x_n$ is defined by

$$x_i(t+1) = f_i\left(\sum_{j=1}^{n} a_{ij} x_j(t) + \sum_{j=1}^{p} b_{ij} u_j(t) + c_i\right)$$

where a_{ij}, b_{ij}, $c_i \in \mathbb{R}$. The u_j's are binary input lines: $u_j(t) \in \{0,1\}$. In the following we assume that there are only two input lines u_1 and u_2. These networks can be used to recognize boolean languages as follows. Starting from the initial state $x(0) = 0$, the input is read sequentially on u_1; u_2 is a validation line which remains active (in state 1) until the whole input has been read. The output is read on two designated output processors, say x_1 and x_2. x_2 is also a validation processor, active only when $x_1(t)$ is the actual result of the computation.

In order to apply the results of the previous sections, we assume that the output function f_i of each processor is a (possibly discontinuous) piecewise-linear function.

THEOREM 15. *If a language $L \in \{0,1\}^*$ can be recognized in polynomial time by a recurrent neural network, then* $L \in \mathrm{P/poly}$.

Proof. This is a straightforward consequence of Theorem 4, since recurrent networks can be simulated with additions, comparisons, and multiplication by constants only. ▮

This theorem can be applied to discontinuous transition functions; this was not the case for the results of [22]. The authors of this paper showed that it is not possible to compute more than P/poly in polynomial time if the f_i's are Lipschitz and polynomial-time approximable (this property holds for most continuous functions of practical interest).

8.2. *Feedforward Networks*

It is shown in [16] that a family of acyclic neural nets of constant depth and polynomial size using piecewise-polynomial activation functions (defined by real parameters) can be simulated by a family of threshold circuits which is still of constant depth and polynomial size. In this section, we consider families of networks of the same type and show that they can be simulated for boolean inputs and outputs by families of threshold circuits with a "small" overhead in

size and depth. In contrast to [16], we can obtain results for networks of more than constant depth ($\log n$ in the worst case and $n^{O(1)}$ in the best) using the additional assumption that the number of distinct real parameters occurring in the networks is bounded by a constant.

DEFINITION 5. A feedforward net is an acyclic graph of gates. Each gate g computes an output of the form

$$y = f_g\left(\sum_{i=1}^{k} w_i x_i - \theta\right), \tag{6}$$

where $x_1, ..., x_k$ are the outputs of the predecessors of g (the parameters $w_1, ..., w_k$, θ are called "weights"). The output functions f_g is piecewise polynomial, i.e., there is a finite sequence $c_0 = -\infty < c_1 < \cdots < c_{l-1} < c_l = +\infty$ such that f is polynomial on each interval $]c_i, c_{i+1}[$. There are n boolean input gates and a single gate of fan-out zero (the output gate). This gate is a threshold gate, i.e., its output function f is such that $f(x) = 1$ if $x \geqslant 0$ and $f(x) = 0$ otherwise.

THEOREM 16. *Let $C = (C_n)_{n \geqslant 1}$ be a family of feedforward networks satisfying the following conditions*:

1. *The family is of polynomial size and $O(\log n)$ depth.*
2. *The number of distinct weights in C_n is bounded by a constant p which is independent of n.*
3. *There exists $d \in \mathbb{N}$ such that for every n and for every output function f_g of C_n, the degrees of the polynomial pieces of f_g are bounded by d.*

Then C can be simulated by a family $(T_n)_{n \geqslant 1}$ of threshold circuits of polynomial size and depth $O(\log n)$. Moreover, the gates of T_n have binary weights (from $\{-1, 1\}$).

Proof Sketch. As in the proof of Theorem 4, we replace for each n the real weights $\alpha_1, ..., \alpha_p$ occuring in C_n by rational approximations $\bar{\alpha}_1, ..., \bar{\alpha}_p$. These approximations must satisfy the constraint that for each input $u \in \{0, 1\}^n$ and each gate g of C_n, the polynomial piece of f_g selected in the computation of C_n on input u is the same as with the actual parameters $\alpha_1, ..., \alpha_p$. It follows from the hypotheses on C that this constraint can be expressed by a system of polynomial inequations of degree and coefficient size $n^{O(1)}$. Hence by Theorem 6, $\bar{\alpha}_1, ..., \bar{\alpha}_p$ can be taken of polynomial size.

We can thus conclude like in [16]: C_n can be simulated by a threshold circuit T_n of polynomial size and depth $O(\log n)$ since the basic arithmetic operations (comparison, multiplication, and multiple addition of integers) can be performed by threshold circuits (with binary weights) of polynomial size and constant depth [6] (here we use the fact that the output functions have uniformly bounded degrees).

Note that in this construction we deal with equality cases with the technique of Section 5 (hence in the complete construction another parameter $\bar{\varepsilon}$ has to be used in addition to $\overline{\alpha_1}, ..., \overline{\alpha_p}$). This avoids the problem of having to perform the algebraic computations of Section 6 with shallow threshold circuits. ∎

If all network gates have piecewise-linear output functions, it is no longer necessary to assume logarithmic depth since there is no explosion of degree or coefficient size in this case (see Lemma 2). Such networks can be obtained for instance by "unwinding" the recurrent networks of Section 8.1.

ACKNOWLEDGMENTS

Several improvements in the presentation and content of this paper were suggested by the anonymous referee, and the presentation of Section 8.2 was suggested by Wolgang Maass. I am greatly indebted to Felipe Cucker for his help with the proof of Lemma 5.

REFERENCES

1. J. L. Balcázar, J. Diáz, and J. Gabarró, "Structural Complexity I," EATCS Monographs on Theoretical Computer Science, Springer-Verlag, New York/Berlin, 1988.
2. J. L. Balcázar, J. Diáz, and J. Gabarró, "Structural Complexity II," EATCS Monographs on Theoretical Computer Science, Springer-Verlag, New York/Berlin, 1990.
3. R. Benedetti and J.-J. Risler, "Real Algebraic and Semi-algebraic Sets," Hermann, Paris, 1990.
4. L. Blum, M. Shub, and S. Smale, On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines, *Bull. Amer. Math. Soc.* **21**, No. 1 (1989), 1–46.
5. J. Bochnak, M. Coste, and M.-F. Roy, "Géométrie algébrique réelle," Springer-Verlag, New York/Berlin, 1987.
6. J. Bruck and K.-Y. Siu, On the power of threshold circuits with small weights, *SIAM J. Discrete Math.* **4**, No. 3 (1991), 423–435.
7. M. Coste and M.-F. Roy, Thom's lemma, the coding of real algebraic numbers and the topology of semi-algebraic sets, *J. Symbolic Comput.* **5** (1988), 121–129.
8. F. Cucker, $\mathrm{P}_{\mathbb{R}} \neq \mathrm{NC}_{\mathbb{R}}$, *J. Complexity* **8** (1992), 230–238.
9. F. Cucker, L. Gonzalez Vega, and F. Roselló, On algorithms for real algebraic plane curves, *in* "Effective Methods in Algebraic Geometry," *Progress in Mathematics*, Vol. 94, pp. 63–87, Birkhäuser, Basel, 1991.
10. F. Cucker, M. Shub, and S. Smale, Separations of complexity classes in Koiran's weak model, *Theoret. Comput. Sci.* **133**, No. 1 (1994), 3–14.
11. J. B. Goode, Accessible telephone directories, *J. Symbolic Logic* **59**, No. 1 (1994), 92–105.
12. C. Kenyon and L. Lóvász, "Algorithmic Discrete Mathematics," Techical Report CS-TR-251-90, Department of Computer Science, Princeton University, March 1990.
13. P. Koiran, A weak version of the Blum, Shub, and Smale model, *in* "Proc. 34th IEEE Symposium on Foundations of Computer Science, 1993," pp. 486–495.

14. S. Lang, "Algebra," Addison-Wesley, Reading, MA, 1965.
15. R. Loos, Computing in algebraic extensions, *in* "Computer Algebra, Symbolic and Algebraic Computation" (Buchberger, Collins, and Loos, Eds.), pp. 173–187, Springer-Verlag, New York/Berlin, 1982.
16. W. Maass, Bounds for the computational power and learning complexity of analog neural nets, *in* "Proc 25th ACM Symposium on Theory of Computing, 1993," pp. 335–344; *SIAM J. Comput.*, to appear.
17. J. Milnor, On the Betti numbers of real varieties, *Proc. Amer. Math. Soc.* **15** (1964), 275–280.
18. J. Renegar, On the computational complexity and geometry of the first-order theory of the reals, parts I, II, III, *J. Symbolic Comput.* **13**, No. 3 (1992), 255–352.
19. F. Ronga, Recherche de solutions d'inéquations polynômiales, *in* "Algorithmique, topologie et géométrie algébriques" (C. Hayat-Legrand and F. Sergeraert, Eds.), Astérisque, Vol. 192, pp.11–16, Soc. Math. de France, Paris, 1990.
20. A. Schönhage, On the power of random access machines, *in* "Proc. 6th ICALP," Lect. Notes in Comput. Sci., Vol. 71, Springer-Verlag, New York/Berlin, 1979.
21. A. Schrijver, "Theory of Linear and Integer Programming, "Wiley, New York, 1986.
22. H. T. Siegelmann and E. D. Sontag, Analog computation via neural networks, *Theoret. Comput. Sci.* **131**, No. 2 (1994), 331–360.
23. S. Smale, Some remarks on the foundations of numerical analysis, *SIAM Rev.* **32**, No. 2 (1990), 211–220.